A SIMPLE ALGORITHM FOR FINDING SHORT SIGMA-DEFINITE REPRESENTATIVES

JEAN FROMENTIN AND LUIS PARIS

ABSTRACT. We describe a new algorithm which for each braid returns a quasi-geodesic σ-definite word representative, defined as a braid word in which the generator σ_i with maximal index i appears either only positively or only negatively.

INTRODUCTION

Since [4], we know that Artin's braid groups B_n are left orderable, by an ordering that enjoys many remarkable properties. This braid ordering is based on the property that every nontrivial braid admits a σ-definite representative, defined to be a braid word in standard Artin generators σ_i in which the generator σ_i with highest index i appears either with only positive exponents or with only negative exponents. In the past two decades, many different proofs of this result have been found, some of them based on algebra [3, 4, 5, 14], other on geometry [2, 8, 9]. All these methods turn out to be algorithms. But in the best cases, starting with a braid word w of length ℓ, they only prove the existence of a σ-definite word w' equivalent to w with length bounded by an exponential on ℓ. In [10], an algorithm returning a quasi-geodesic σ-definite representative has been introduced. It is heavily based on technical properties of the so-called rotating normal form on the Birman–Ko–Lee monoid. Quite effective, this algorithm remains complicate.

The aim of this paper is to describe a simple algorithm returning a quasi-geodesic σ-definite representative. It is based either on the alternating normal form introduced in [6] or on the rotating normal form intoduced in [10, 11, 12]. The main advantage of this new algorithm is that it can be describe with few technical results on these normal forms. Part of the algorithm presented here uses some ideas from [10]. However, this new algorithm goes beyond the simplification of the previous one, and the paper can be read independently from [10].

The paper is organized as follows. In Section 1, we give an overview on reversing processes and give some elementary algorithm that will be needed to describe the main algorithm. In Section 2 we recall the definition of the Φ_n-splitting, that is a natural way to describe each braid of B_n^+ from a finite sequence of braid of B_{n-1}^+, and we give an algorithm to compute it. In Section 3, we introduce two different algorithms that allow us to express a braid of B_n as a quotient of braids lying in B_n^+. In Section 4 we describe and prove the correctness of the main algorithm

Date: 14 décembre 2010.

2010 *Mathematics Subject Classification.* 20F36,20M05,06F05.

Key words and phrases. Braid group, braid monoids, braid ordering, algorithm.

Both authors are partially supported by the *Agence Nationale de la Recherche* (*projet Théorie de Garside*, ANR-08-BLAN-0269-03).

in the context of the alternating normal form. Finally, in the last section, we investigate the complexity of our algorithm in the context of the Birman–Ko–Lee monoid B_n^{+*}.

1. Reversing process

In this section, we recall how to perform elementary computations in a finitely generated Garside monoid. The main tool is the reversing algorithm introduced in [7].

Assume that M is a Garside monoid. Then, we define two partial orderings on M. Given elements β and β' of M, we say that β *left divides* (resp. *right divides*) β', denoted by $\beta \prec \beta'$ (resp. $\beta \succ \beta'$), if there exists γ in M such that $\beta\,\gamma = \beta'$ (resp. $\beta = \gamma\,\beta'$) is satisfy.

The *left lcm* of two elements β and β' of M is the minimal element γ in M, with respect to $\prec$, satisfying $\beta \prec \gamma$ and $\beta' \prec \gamma$, and we denote it by $\beta \vee_{L} \beta'$. Of course, we define symmetrically the *right lcm* of β and β' in M, which is denoted by $\beta \vee_{R} \beta'$.

Definition 1.1. Let M be a monoid generated by a finite set S.

(i) A word on the alphabet S is called a *positive S-word*,

(ii) A word on the alphabet $S \cup S^{-1}$ is called a *S-word*,

(iii) The element represented by an S-word w is denoted by $\overline{w}$,

(iv) For w, w' two S-words, we say that w is equivalent to w', denoted by $w \equiv w'$, if $\overline{w} = \overline{w'}$ holds.

Let M be a Garside monoid generated by a finite set S. A *left lcm selector* on S in M is a mapping $f_L^n : S \times S \to S^*$ such that, for all x, y in S, the words $x\, f_L^n(x,y)$ and $y\, f_L^n(y,x)$ both represent $\overline{x} \vee_L \overline{y}$. We define also a *right lcm selector* on S in M to be a mapping f_R^n such that $f_R^n(x,y)\, y$ and $f_R^n(y,x)\, x$ represent $\overline{x} \vee_R \overline{y}$ for all x, y in S.

Example 1.2. We recall that the positive braid monoid B_n^+ is defined for $n \geqslant 2$ by the presentation

$$\left\langle \sigma_1, \dots, \sigma_{n-1};\ \begin{array}{rcll} \sigma_i\sigma_j &=& \sigma_j\sigma_i & \text{for } |i-j| \geqslant 2 \\ \sigma_i\sigma_j\sigma_i &=& \sigma_j\sigma_i\sigma_j & \text{for } |i-j| = 1 \end{array} \right\rangle^{+}. \tag{1}$$

We put $\Sigma_n = \{\sigma_1, \dots, \sigma_{n-1}\}$. Then the applications f_L^n and f_R^n defined on $\Sigma_n \times \Sigma_n$ by

$$f_L^n(\sigma_i, \sigma_j) = f_R^n(\sigma_i, \sigma_j) = \begin{cases} \sigma_j & \text{for } |i-j| \geqslant 2, \\ \sigma_j\sigma_i & \text{for } |i-j| = 1. \end{cases}$$

are respectively left and right lcm selectors on Σ_n in B_n^+.

For the rest of this section, we fix a Garside monoid M, a finite generating set S of M, a left lcm selector f_L^n and a right lcm selector f_R^n on S in M.

Definition 1.3. Let w, w' be S-words. We say that $w \curvearrowright_R^{(1)} w'$ is true if w' is obtained from w by replacing a subword $x^{-1}y$ of w by $f_L^n(x,y)\, f_L^n(y,x)^{-1}$. We say that $w \curvearrowright_R w'$ is true if there exists a sequence $w = w_0, \dots, w_k = w'$ of S-words such that $w_i \curvearrowright_R^{(1)} w_{i+1}$ holds for all $i = 0, 1, \dots, k-1$.

Symmetrically, we say that $w \curvearrowright_L w'$ is true, if w' is obtained form w by repeatedly replacing a subword $x\,y^{-1}$ of w by the word $f_R^n(y,x)^{-1}\, f_R^n(x,y)$.

We now introduce the notion of right reversing diagrams. Assume that $w_0, \dots, w_k$ is a reversing sequence, *i.e.*, a sequence of S-words such that $w_i \curvearrowright_R^{(1)} w_{i+1}$ holds for each i. First we associate with w_0 a path labelled with its successive letters: we associate to a positive letter x a horizontal

right-oriented arrow labelled x, and to a negative letter x^{-1} a vertical down-oriented arrow labelled x. Then we successively represent the S-words $w_1, \ldots, w_k$ as follows: if w_{i+1} is obtained form w_i by replacing the subword $x^{-1}y$ of w_i by $f_L^n(x,y)\, f_L^n(y,x)^{-1}$, then we complete the pattern corresponding to $x^{-1}y$ using a right-oriented arrow labelled $f_L^n(x,y)$ and a down-oriented arrow labelled $f_L^n(y,x)$ to obtain a square:

$$\begin{tikzcd} \cdot \arrow[r, "y"] \arrow[d, "x"'] & \cdot \\ \cdot & \end{tikzcd} \quad \text{completed into} \quad \begin{tikzcd} \cdot \arrow[r, "y"] \arrow[d, "x"'] \arrow[dr, phantom, "\curvearrowright_R"] & \cdot \arrow[d, dashed, "f_L^n(y,x)"] \\ \cdot \arrow[r, dashed, "f_L^n(x,y)"'] & \cdot \end{tikzcd}$$

Symmetrically, we define a left reversing diagram, in which we complete the pattern corresponding to xy^{-1} using a right-oriented arrow labelled $f_R^n(x,y)$ and a down-oriented arrow labelled $f_R^n(y,x)$:

$$\begin{tikzcd} & \cdot \arrow[d, "y"] \\ \cdot \arrow[r, "x"'] & \cdot \end{tikzcd} \quad \text{completed into} \quad \begin{tikzcd} \cdot \arrow[r, dashed, "f_R^n(x,y)"] \arrow[d, dashed, "f_R^n(y,x)"'] \arrow[dr, phantom, "\curvearrowright_L"] & \cdot \arrow[d, "y"] \\ \cdot \arrow[r, "x"'] & \cdot \end{tikzcd}$$

Proposition 1.4. [7] *For every S-word w, there exist unique positive S-words u and v such that $w \curvearrowright_R u\, v^{-1}$ holds. Moreover the words u and v are obtained from w in time $O(pos(w) \cdot neg(w))$, where $pos(w)$ is the number of positive letters occurring in w and $neg(w)$ is the number of negative letters occurring in w. A similar result occurs for $\curvearrowright_L$.*

Let w be a S-word. As there exist unique positive S-words u, v such that $w \curvearrowright_R uv^{-1}$ holds, we say that u is the *right numerator* of w, denoted by $N_R(w)$, and that v is the *right denominator* of w, denoted by $D_R(w)$. Symmetrically, we define *left numerator* and *left denominator* of w respectively denoted by $N_L(w)$ and $D_L(w)$. An immediate consequence of [7] is:

Proposition 1.5. *For all positive S-words u, v:*
(i) $\overline{u} \prec \overline{v}$ holds if and only if $D_R(u^{-1}v)$ is the empty word ε,
(ii) $\overline{u} \succ \overline{v}$ holds if and only if $D_L(uv^{-1})$ is the empty word ε.

Let β, β' be two elements of M. The *left gcd* of β and β' is the maximal element γ with respect to $\prec$ such that $\gamma \prec \beta$ and $\gamma \prec \beta'$ holds.

Proposition 1.6. [7, Proposition 7.7] *Let u, v be positive S-words. Then the left gcd of $\overline{u}$ and $\overline{v}$ is the element represented by*

$$N_L(uD_L(N_R(u^{-1}v)D_R(u^{-1}v)^{-1})^{-1}). \tag{2}$$

See Figure 1 for a description of (2) in terms of reversing diagrams.

The reversing process takes a word on input and returns a word. In order to simplify notations we shall use divisor and gcd symbols on words.

Notation 1.7. Let u, v be positive S-words.
(i) If $\overline{u} \prec \overline{v}$ holds, we denote by $u\backslash v$ the word $N_R(u^{-1}\, v)$.
(ii) If $\overline{u} \succ \overline{v}$ holds, we denote by u/v the word $N_L(u\, v^{-1})$.
(iii) We denote by $u \wedge_L v$ the word of (2).

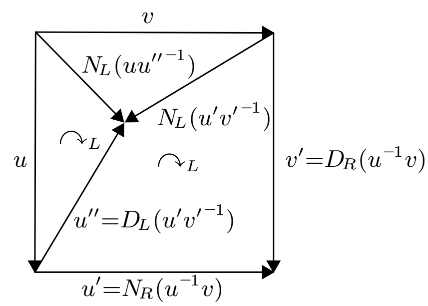

FIGURE 1. Reversing diagram corresponding to the computation of the left gcd of $\overline{u}$ and $\overline{v}$. Firstly, we right reverse $u^{-1}v$ to obtain $u'v'^{-1}$. Secondly we left reverse $u'v'^{-1}$ to compute $D_L(u'v'^{-1})$, denoted by u''. Finally we left reverse uu''^{-1} to compute $N_L(uu''^{-1})$, which represents $\overline{u} \wedge_L \overline{v}$.

With these notations, the element $\overline{u\backslash v}$ is equal to $\overline{u}^{-1}\overline{v}$, the element $\overline{u/v}$ is equal to $\overline{u}\,\overline{v}^{-1}$ and the element $\overline{u \wedge_L v}$ is the left gcd of $\overline{u}$ and $\overline{v}$.

In the sequel we will consider two Garside monoids naturally generated by a finite set, namely the positive braid monoid B_n^+ generated by Σ_n and the dual braid monoid B_n^{+*} generated by A_n (see Section 5). From now on, we will not specify the lcm selectors for left and right reversing operations in these monoids, if not needed.

2. The Φ_n-splitting

It is shown in [6] how associate with every braid β of B_n^+ a unique sequence of braids in B_{n-1}^+, called the Φ_n*-splitting* of β, that completely determines β. As mentioned in the introduction, our algorithm is based on this operation. In this section we recall the definition and the construction of the Φ_n-splitting of a braid.

We recall that the positive braid monoid B_n^+ is a Garside monoid with Garside element Δ_n defined by

$$\Delta_n = (\sigma_1 \dots \sigma_{n-1}) \cdot (\sigma_1 \dots \sigma_{n-2}) \cdot \dots \cdot (\sigma_1\, \sigma_2) \cdot \sigma_1.$$

See [7, 13] for a definition of a Garside monoid.

We denote by Φ_n the *flip automorphism* of B_n^+, *i.e.*, the application defined on B_n^+ by $\Phi_n(\beta) = \Delta_n\, \beta\, \Delta_n^{-1}$. The initial observation of the construction of the alternating normal form is that each braid of B_n^+ admits a unique maximal right divisor lying in $\Phi_n^k(B_{n-1}^+)$ for all k.

Lemma 2.1. *For $n \geqslant 3$ and $k \geqslant 0$, every braid β of B_n^+ admits a unique maximal right divisor β_1 lying in $\Phi_n^k(B_{n-1}^+)$.*

Proof. The braid β_1 is a maximal right divisor of β lying in $\Phi_n^k(B_{n-1}^+)$ if and only if $\Phi_n^{-k}(\beta_1)$ is the maximal right divisor of $\Phi_n^{-k}(\beta)$ lying in B_{n-1}^+. As the submonoid B_{n-1}^+ of B_n^+ is closed under right divisors and left lcm, the braid $\Phi_n^{-k}(\beta_1)$ exists and is unique. □

Definition 2.2. The braid β_1 of Lemma 2.1 is called the $\Phi_n^k(B_{n-1}^+)$-tail of β.

By iterating the tail construction, we then associate with every braid of B_n^+ a finite sequence of braids of B_{n-1}^+.

Proposition 2.3. [6, Proposition 2.5] *Assume $n \geqslant 3$. Then for each nontrivial braid β of B_n^+, there exists a unique sequence $(\beta_b, \dots, \beta_1)$ in B_{n-1}^+ satisfying*

$$\beta_b \neq 1 \quad \textit{and} \quad \beta = \Phi_n^{b-1}(\beta_b) \cdot \dots \cdot \Phi_n(\beta_2) \cdot \beta_1 \tag{3}$$

$$\textit{for each } k \geqslant 1,\ \Phi_n^{k-1}(\beta_k) \textit{ is the } \Phi_n^{k-1}(B_{n-1}^+)\textit{-tail of } \Phi_n^{b-1}(\beta_b) \cdot \dots \cdot \Phi_n^{k-1}(\beta_k) \tag{4}$$

Definition 2.4. The sequence $(\beta_b, \dots, \beta_1)$ of Proposition 2.3 is called the Φ_n-splitting of β and its length is called the Φ_n-breadth of β.

We give now an algorithm to compute the Φ_n-splitting of a braid given by a positive Σ_n-word w. More precisely the algorithm returns a sequence $(w_b, \dots, w_1)$ of positive Σ_{n-1}-words such that $(\overline{w}_b, \dots, \overline{w}_1)$ is the Φ_n-splitting of $\overline{w}$.

Algorithm 1. Compute the Φ_n-splitting of the braid represented by w
Input: A positive Σ_n-word w with $n \geqslant 3$
1. `Put` ß $= (\,)$, $w' = w$ and $k = 0$.
2. `While` $w' \neq \varepsilon$ `do`
3.   `Put` $u = \varepsilon$.
4.   `While` there exists $x \in \Sigma_{n-1}$ such that $\overline{w'} \succ \Phi_n^k(\overline{x})$ `do`
5.   `Put` $w' = w'/\Phi_n^k(x)$ and $u = x\,u$
6.   Insert u on the left of ß.
7.   `Put` $k = k+1$
8. `Return` s.

Proposition 2.5. *Running on w, Algorithm 1 ends in time $O(|w|^2)$ and returns a sequence $(w_b, \dots, w_1)$ of positive Σ_{n-1}-words such that $(\overline{w}_b, \dots, \overline{w}_1)$ is the Φ_n-splitting of $\overline{w}$.*

Proof. We denote by β the braid represented by the value of w' at Line `3`. Lines `3`, `4` and `5` compute the maximal right divisor of β lying in $\Phi_n^k(B_{n-1}^+)$. At Line `6`, the braid $\Phi_n^k(\overline{u})$ is equal to the $\Phi_n^k(B_{n-1}^+)$-tail of β and $\overline{w'}$ is equal to $\beta/\Phi_n^k(\overline{u})$.

Therefore the algorithm applies successively the $\Phi_n^k(B_{n-1}^+)$-tail construction for $k = 1, 2, \dots$. Then, by Proposition 2.3 it must stop and return the expected sequence of words.

As for time complexity, testing if $\overline{w'} \succ \Phi_n^k(\overline{x})$ holds and computing $w'/\Phi_n^k(x)$ need to run the left revering process on $w'\,(\Phi_n^k(x))^{-1}$. Proposition 1.4 guarantees that these two operations can be done in time $O(|w'|)$, and so, in time $O(|w|)$ since $|w'| \leqslant |w|$ holds. Then an easy bookkeeping shows that the algorithm ends in time $O(|w|^2)$. □

3. Garside quotient

In the previous section we have seen how to compute the Φ_n-splitting of a braid lying in B_n^+. Of course there is no possible extension of the notion of Φ_n-splitting to the braid group B_n. However, we have the following.

Proposition 3.1. *Each braid β admits a unique decomposition $\Delta_n^{-t}\,\beta'$ where t is a nonnegative integer and β' is a braid belonging to B_n^+, which is not left divisible by Δ_n, unless $t = 0$.*

Proof. The monoid B_n^+ is a Garside monoid with Garside elements Δ_n, see [13]. As B_n is the group of fractions of B_n^+, there exist a smallest nonnegetive integer t such that $\Delta_n^t\,\beta$ lies in B_n^+.

If t is positive, the minimality hypothesis on t implies $\Delta_n \not\prec \Delta_n^t\,\beta$. Then we define β' to be the braid $\Delta_n^t\,\beta$.

Assume now that $\Delta_n^{-t'}\,\beta''$ is another decomposition of β satisfying the hypothesis of the proposition. As $\Delta_n^{t'}\beta$ belongs to B_n^+, we have $t' \geqslant t$. Assume $t' > t$. Then we have $t' > 0$. As the braid β'' is equal to $\Delta_n^{t'-t}\,\beta'$, the relation $\Delta_n \prec \beta''$ holds, that is in contradiction with $t' > 0$ and the hypothesis of the proposition. Hence t' is equal to t and then β' is equal to β''. $\square$

Algorithm 2. Compute the decomposition $\Delta_n^{-t}\,\overline{v}$ given in Proposition 3.1 of the braid represented by the Σ_n-word w.

Input: An Σ_n-word w

1. Write w as $w_0\,x_1^{-1}\,w_1 \,...\, w_{t-1}\,x_t^{-1}\,w_t$ (where w_i is a positive word and x_j is a letter).
3. `For` $i = 1 \,...\, t$ compute u_i such that $\Delta_n = u_i\,x_i$.
4. `Put` $v = \Phi_n^t(w_0)\,\Phi_n^{t-1}(u_1\,w_1)\,...\,\Phi_n(u_{t-1}\,w_{t-1})\,u_t\,w_t$.
5. `While` $\Delta_n \prec \overline{v}$ and $t > 0$ hold `do`
6.   `Put` $v = \Delta_n\backslash v$ and $t = t-1$.
7. `Return` $\Delta_n^{-t}\,v$.

Proposition 3.2. *Running on w, Algorithm 2 ends in time $O(|w|^2)$ and has the correct output. Moreover we have $|\Delta_n^{-t}u| \leqslant (n^2{-}n{-}1)\cdot\|\overline{w}\|_\sigma$, where $\|\beta\|_\sigma$ is the minimal length of a Σ-word representing β.*

Proof. For $i = 1, \ldots, t$, we denote by x_i^{-1} the negative letters occurring in w. Then we replace each x_i^{-1} by $\Delta_n^{-1}\,u_i$ to obtain

$$w \equiv w_0\,\Delta_n^{-1}\,u_1\,w_1\,...\,w_{t-1}\,\Delta_n^{-1}\,u_t\,w_t. \tag{5}$$

The definition of Φ_n implies $u\,\Delta_n^{-1} \equiv \Delta_n^{-1}\,\Phi_n(u)$ for every positive Σ_n-word u. From relation (5), we obtain

$$w \equiv \Delta_n^{-t}\Phi_n^t(w_0)\,\Phi_n^{t-1}(u_1\,w_1)\,...\,\Phi_n(u_{t-1}\,w_{t-1})\,u_t\,w_t. \tag{6}$$

So the word v introduced in Line 4 is equivalent to $\Delta_n^t\,w$. After Lines 5 and 6, the braid v is not left divisible by Δ_n unless $t = 0$. At the end, we have $w \equiv \Delta_n^{-t}\,v$, hence the algorithm returns the correct output.

As for the length, replacing x_i^{-1} by $\Delta_n^{-1}\,u_i$ multiplies it by at most $2|\Delta_n| - 1$, *i.e.*, by at most $n^2{-}n{-}1$. Indeed, the relations in the presentation (1) preserve the length, hence we have $|u_i| = |\Delta_n|{-}1$. By Proposition 3.1, the integer t and the braid $\overline{v}$ depend only of the braid $\overline{w}$ and not on the word w. Hence, applying the algorithm to a geodesic word representing $\overline{w}$ gives

$$|\Delta_n^{-t}v| \leqslant (n^2{-}n{-}1)\cdot\|\overline{w}\|_\sigma.$$

As for time complexity, the word of Line 4 is obtained in time $O(|w|)$. The `while` command of Line 5 needs at most $|v|$ steps. Testing if $\Delta_n \prec \overline{v}$ holds and computing $\Delta_n\backslash v$ need to run the right reversing process on $\Delta_n^{-1}\,v$. Proposition 1.4 guarantees that these two operations can be done in time $O(|v|)$. Then, from $|v| \leqslant (n^2{-}n{-}1)\cdot|w|$, we deduce that the algorithm ends in time $O(|w|)^2$. $\square$

Now, for a braid β, the decomposition which we shall introduce in the next proposition is called the Garside–Thurston normal form of β. We use this normal form for computing the minimal k such that β lies in B_k. Indeed, as we will see, the Garside-Thurston normal form depends only on β and not on the group B_n in which it is viewed.

Proposition 3.3. [7, Corollary 7.5] *Each braid β of B_n admits a unique decomposition ${\beta'}^{-1}\,\beta''$ where β', β'' belong to B_n^+ and such that $\beta' \wedge_L \beta''$ is trivial. Moreover if β is represented by w then the braid β' is represented by $D_L(N_R(w)D_R(w)^{-1})$ and the braid β'' is represented by $N_L(N_R(w)D_R(w)^{-1})$.*

Since $for\ k \leqslant n$ the lattice operation $\wedge_L$ in B_k^+ coincides with that of B_n^+, a direct consequence of Proposition 3.3 is the following.

Corollary 3.4. *Let $k \leqslant n$, β in B_n and $\beta = {\beta'}^{-1}\beta''$ be the Garside-Thurston normal form of β. We have $\beta \in B_k$ if and only if β', β'' lie in B_k^+.*

Definition 3.5. We define the *index* of a Σ_n-word w to be the maximal i such that w contains a letter σ_{i-1}. The *index* of a braid β is the minimal index of a word which represents β.

Obviously, the index of a braid β is the minimal integer n such that β lies in B_n.

Algorithm 3. Compute the index k of $\overline{w}$ and a Σ_k-word w'' equivalent to w.
 Input: An Σ_n-word w
 1. Right reverse w into w'.
 2. Left reverse w' into w''.
 3. Let k be the index of w''.
 4. `Return` (k, w'').

The correctness of this algorithm is a direct consequence of Proposition 3.3 together with Corollary 3.4. Moreover, by Proposition 1.4 it ends in time $O(|w|^2)$.

4. The main algorithm

Putting all pieces together, we can now describe our algorithm which returns a quasi-geodesic word equivalent to a given word. However, we first recall the definition of σ-definite words and give the result of [6] which will be used to prove the correctness of the algorithm. As ever we assume $\Sigma_n \subset \Sigma_{n+1}$ (as well as $B_n \subset B_{n+1}$) for all $n \geqslant 2$, and we set $\Sigma = \bigcup_{n=2}^{\infty} \Sigma_n$.

Definition 4.1.

(i) A Σ-word is said to be σ_i-positive (resp. σ_i-negative) if it contains at least one letter of the form σ_i, no letter σ_i^{-1} (resp. at least one letter σ_i^{-1}, no letter σ_i) and no letter σ_j with $j > i$.

(ii) A Σ-word is said to be σ-definite if it is either trivial, or σ_i-positive or σ_i-negative for a certain i.

(iii) A braid is said to be σ_i-positive (resp. σ_i-negative) if it can be represented by a σ_i-positive word (resp. a σ_i-negative word).

Recall form [4] that the celebrated Dehornoy ordering on B_n is defined by $\beta < \gamma$ if $\beta^{-1}\gamma$ is σ_i-positive for some $i \leqslant n$. The key property that will be used on the Φ_n-splitting operation is its coincidence with the Dehornoy ordering $<$.

Proposition 4.2. [6] *Let β and γ be two braids of B_n^+. Let $(\beta_b, \dots, \beta_1)$ and $(\gamma_c, \dots, \gamma_1)$ be the Φ_n-splittings of β and γ respectively. Then $\beta < \gamma$ holds if and only if we have either $b < c$ or $b = c$ and, for some $t \leqslant b$, we have $\beta_{t'} = \gamma_{t'}$ for $t < t' \leqslant b$ together with $\beta_t < \gamma_t$.*

In [6], Dehornoy proves that the minimal positive braid of a given Φ_n-breadth $b+2$ (see 2.4) with $b \geqslant 0$ is $\widehat{\Delta}_{n-1,b} = \Delta_n^b\,\Delta_{n-1}^{-b}$, and the Φ_n-splitting of the latter is the following sequence of

length $b+2$

$$(\sigma_1, \sigma_{n-1} \dots \sigma_2\, \sigma_1^2, \dots, \sigma_{n-1} \dots \sigma_2\, \sigma_1^2, \sigma_{n-1} \dots \sigma_2\, \sigma_1, 1). \tag{7}$$

As the braid Δ_{n-1} lies in B_{n-1}^+, we deduce that the Φ_n-splitting of Δ_n^b is the following sequence of length $b+2$

$$(\sigma_1, \sigma_{n-1} \dots \sigma_2\, \sigma_1^2, \dots, \sigma_{n-1} \dots \sigma_2\, \sigma_1^2, \sigma_{n-1} \dots \sigma_2\, \sigma_1, \Delta_{n-1}^{b-1}). \tag{8}$$

The idea of our algorithm is that we can easily decide if a quotient $\Delta_n^{-t}\,\beta$ with β lying in B_n^+ is σ_{n-1}-negative or not.

Lemma 4.3. *Assume that β is a braid of B_n^+ such that $\widehat{\Delta}_{n-1,b} \leqslant \beta \leqslant \Delta_n^b$ holds, then the quotient $\Delta_n^{-b}\beta$ lies in B_{n-1}.*

Proof. The relation $\widehat{\Delta}_{n-1,b} \leqslant \beta \leqslant \Delta_n^b$ and Proposition 4.2 imply that the Φ_n-splitting of β is the following sequence of length $b+2$

$$(\sigma_1, \sigma_{n-1} \dots \sigma_2\, \sigma_1^2, \dots, \sigma_{n-1} \dots \sigma_2\, \sigma_1^2, \sigma_{n-1} \dots \sigma_2\, \sigma_1, \beta_1), \tag{9}$$

with $1 \leqslant \beta_1 \leqslant \Delta_{n-1}^b$. Hence β is equal to $\widehat{\Delta}_{n-1,b}\,\beta_1$ where β_1 belongs to B_{n-1}^+. Then, as the quotient $\Delta_n^{-b}\,\beta$ is equal to $\Delta_{n-1}^{-b}\,\widehat{\Delta}_{n-1,b}\,\beta$, we obtain $\Delta_n^{-b}\,\beta = \Delta_{n-1}^{-b}\,\beta_1$. As β_1 and Δ_{n-1} lie in B_{n-1} the braid $\Delta_n^{-b}\,\beta$ lies in B_{n-1}. □

Proposition 4.4. *Assume $n \geqslant 3$ and β is a braid of B_n^+. Let t be a positive integer and b the Φ_n-breadth of β. If $t \geqslant b-1$ holds then the quotient $\Delta_n^{-t}\,\beta$ is σ_{n-1}-negative. Otherwise it is not σ_{n-1}-negative.*

Proof. Let $(\beta_b, \dots, \beta_1)$ be the Φ_n-splitting of β. Then the braid $\Delta_n^{-t}\beta$ is equal to

$$\Delta_n^{-t} \cdot \Phi_n^{b-1}(\beta_b) \cdot \dots \cdot \Phi_n(\beta_2) \cdot \beta_1. \tag{10}$$

Pushing $b-1$ powers of Δ_n to the right in (10) and dispatching them between the factors β_k, we find

$$\begin{aligned}
\Delta_n^{-t}\beta &\equiv \Delta_n^{-t} \cdot \Phi_n^{b-1}(\beta_b) \cdot \dots \cdot \Phi_n(\beta_2) \cdot \beta_1 \\
&\equiv \Delta_n^{-t+b-1} \cdot \Delta_n^{-b-1}\Phi_n^{b-1}(\beta_b) \cdot \dots \cdot \Phi_n(\beta_2) \cdot \beta_1 \\
&\equiv \Delta_n^{-t+b-1} \cdot \beta_b \cdot \Delta_n^{-1} \cdot \Delta_n^{-b-2} \cdot \dots \cdot \Phi_n(\beta_2) \cdot \beta_1 \\
&\equiv \dots \equiv \Delta_n^{-t+b-1}\ \beta_b\, \Delta_n^{-1}\ \beta_{b-1}\, \Delta_n^{-1} \dots \beta_2\, \Delta_n^{-1}\ \beta_1.
\end{aligned}$$

If the relation $t \geqslant b-1$ holds then the braid

$$\Delta_n^{-t+b-1}\ \beta_b\, \Delta_n^{-1}\ \beta_{b-1}\, \Delta_n^{-1} \dots \beta_2\, \Delta_n^{-1}\ \beta_1, \tag{11}$$

is σ_{n-1}-negative. Indeed, by definition, the braid Δ_n^{-1} is σ_{n-1}-negative, while for each k the braid β_k lies in B_{n-1}^+. So, as $-t+b-1$ is nonpositive, the expression (11) contains t letters σ_{n-1}^{-1} and no letter σ_{n-1}.

Now assume $t < b-2$. The Φ_n-breadth of Δ_n^t is $t+2$ and we have $t+2 < b$. Then Proposition 4.2 implies $\Delta_n^t < \beta$, *i.e.*, $\Delta_n^{-t}\,\beta$ is σ_i-positive for a certain i, hence it is not σ_{n-1}-negative.

Finally assume $t = b-2$. If the relation $\Delta_n^t < \beta$ holds we concluse as in the previous case. Then assume $\beta \leqslant \Delta_n^t$. As the Φ_n-breadth of β is b, which is equal to $t+2$, Proposition 4.2 implies $\widehat{\Delta}_{n-1,t} \leqslant \beta$. Then by Lemma 4.3 the quotient $\Delta_n^{-t}\,\beta$ lies in B_{n-1}, hence it is not σ_{n-1}-negative. □

The following algorithm takes in entry a braid word w representing a braid β and it returns a σ-definite word representing β. The main idea is to bring all possible cases to the case where β is σ_{n-1}-negative, *i.e.*, when β satisfy the conditions of Proposition 4.4.

Algorithm 4. Compute a σ-definite representative.

Input: A Σ_n-word w

1. `Put` $e = 1$.
2. Let (k, v) be the output of Algorithm 3 applied to w.
3. Use Algorithm 2 to compute $\Delta_k^{-t}\, u \equiv v^e$.
4. `If` $t = 0$ or $k = 2$ `then return` $\Delta_k^{-t}\, u$.
5. Use Algorithm 1 to compute the Φ_k-splitting $(u_b, \dots, u_1)$ of u.
6. `If` $t \geqslant b - 1$ `then return` $(\Delta_k^{-t+b-1}\ u_b\, \Delta_k^{-1}\ u_{b-1}\, \Delta_k^{-1} \dots u_2\, \Delta_k^{-1}\ u_1)^e$.
7. `Else put` $e = -1$ and `goto Line 3`.

Proposition 4.5. *Algorithm 4 ends and returns in time $O(|w|)^2$ a σ-definite word w' equivalent to w with $|w'| \leqslant (n^2-n-1)\cdot\|\overline{w}\|_\sigma$, where $\|\beta\|_\sigma$ is the minimal length of a Σ-word representing β.*

Proof. We use Algorithm 3 to compute the index k of $\overline{w}$ and a Σ_k-word v equivalent to w. In particular, the braid $\overline{w}$ is either σ_{k-1}-positive or σ_{k-1}-negative.

Next, we use Algorithm 2 to compute a quotient $\Delta_k^{-t}\, u$ that is equivalent to v and so to w. By Proposition 3.1 the exponent t and the positive braid $\overline{u}$ depends only of the braid $\overline{w}$ and not on the word v. Moreover, we have

$$|\Delta_k^{-t} u| \leqslant (k^2-k-1)\|\overline{w}\|_\sigma \leqslant (n^2-n-1)\|\overline{w}\|_\sigma.$$

If t is equal to 0 then the quotient $\Delta_k^{-t}u$ is equal to u, that is a positive word, hence a σ-definite word. If n is equal to 2 with $t \neq 0$ then u is empty and $\Delta_k^{-t}u$ is equal to Δ_k^{-t}, that is a negative word, hence a σ-definite word.

Next, we use Algorithm 1 to compute the Φ_k-splitting $(\overline{u}_b, \dots, \overline{u}_1)$ of $\overline{u}$. Then the word $\Delta_k^{-t}u$ is equivalent to u' defined by

$$u' = \Delta_k^{-t+b-1}\ u_b\, \Delta_k^{-1}\ u_{b-1}\, \Delta_k^{-1} \dots u_2\, \Delta_k^{-1}\ u_1. \tag{12}$$

If the relation $t \geqslant b - 1$ holds then the word u' is σ_{k-1}-negative—see proof of Proposition 4.4—hence it is σ-definite. So in this case the algorithm returns a σ-definite word equivalent to w.

Now, assume $t < b - 1$. In this case we redo the same process with the word w^{-1}. Note that, as the index of w and w^{-1} are the same, we can directly go to Line `3` of the algorithm. In this case, by Proposition 4.4, the braid $\overline{w}$ is not σ_{k-1}-negative, *i.e.*, it is σ_{k-1}-positive or it lies in B_m with $m < k$. As k is the index of $\overline{w}$, the braid $\overline{w}$ is σ_{k-1}-positive. So the braid represented by v^{-1} is σ_{k-1}-negative. Hence the new value of t and b satisfy the relation $t \geqslant b - 1$ and the algorithm ends.

For length complexity, the length of the Σ_k-word u' given in (12) is equal to the length of the Σ_k-word $\Delta_k^{-t}\, u$. By Proposition 3.1 we have

$$\Delta_k^{-t}\, u \leqslant (k^2-k-1)\cdot\|\overline{v}^e\|_\sigma = (k^2-k-1)\cdot\|\overline{w}^e\|_\sigma \leqslant (n^2-n-1)\cdot\|\overline{w}^e\|_\sigma.$$

Then, $\|\overline{w}^{-1}\|_\sigma = \|\overline{w}\|_\sigma$ implies $|u'| \leqslant (n^2-n-1)\|\overline{w}\|_\sigma$. $\square$

5. Dual braid monoid

The *dual braid monoid* is another submonoid of B_n. It is generated by a subset of B_n that properly contains $\{\sigma_1, \dots, \sigma_{n-1}\}$, and consists of the so-called *Birman-Ko-Lee generators* introduced in [1].

Definition 5.1.
(i) For $1 \leqslant p < q$, we put $a_{p,q} = \sigma_p \dots \sigma_{q-2}\ \sigma_{q-1}\ \sigma_{q-2}^{-1} \dots \sigma_p^{-1}$.
(ii) For $n \geqslant 2$, the set A_n is defined to be $\{a_{p,q} \mid 1 \leqslant p < q \leqslant n\}$.
(iii) The *dual braid monoid* B_n^{+*}, is the submonoid of B_n generated by A_n.

For $p < q$, we denote by $[\![p, q]\!]$ the interval $\{p, \dots, q\}$ of $\mathbb{N}$, and we say that $[\![p, q]\!]$ is nested in $[\![r, s]\!]$ if we have $r < p < q < s$. A presentation of B_n^{+*} in terms of $a_{p,p}$ is as follows.

Proposition 5.2. [1] *In terms of the $a_{p,q}$, the monoid B_n^{+*} is presented by*

$$a_{p,q}\, a_{r,s} = a_{r,s}\, a_{p,q} \quad \text{for } [\![p, q]\!] \text{ and } [\![r, s]\!] \text{ disjoint or nested},$$
$$a_{p,q}\, a_{q,r} = a_{q,r}\, a_{p,r} = a_{p,r}\, a_{p,q} \quad \text{for } 1 \leqslant p < q < r \leqslant n.$$

As the positive braid monoid, we can endow the Birman–Ko–Lee monoid B_n^{+*} with a Garside structure. The corresponding Garside element is

$$\delta_n = a_{1,2}\, a_{2,3}\ \dots\ a_{n-1,n}.$$

We denote by ϕ_n the Garside automorphism of B_n^{+*}, *i.e.*, the application defined on B_n^{+*} by $\phi_n(\beta) = \delta_n\, \beta\, \delta_n^{-1}$.

An analog of the alternating normal form of the positive braid monoid B_n^+ exists for the dual braid monoid B_n^{+*}: the *rotating normal form* (see [11, 10] for more details about this normal form). The rotating normal form is also based on an operation of splitting: the ϕ_n-splitting. Moreover, for each result on the alternating normal form used in this paper there exists a counterpart in the B_n^{+*}context, see [11] or [12] for more details.

A A_n-word w is said to be σ-definite if all the letters $a_{p,q}^{\pm}$ with highest q appear only positively or only negatively. This definition coincides with that given for Σ_n-words if we translate each letter $a_{p,q}$ of an A_n-word to the Σ_n-word given in Definition 5.1 (i). Hence all the previous algortihms can be translated to the dual language, replacing Φ_n by ϕ_n, Δ_n by δ_n and Σ_n by A_n. One of the advantage of the dual braid monoid is that its Garside element δ_n has length $n-1$, while Δ_n has length $\frac{n(n-1)}{2}$. Therefore in the dual context, Algorithm 2 runing on w returns a word $\delta_n^{-t}\, u$ whose length is at most $(2n-3)\|\overline{w}\|_A$,where, for $\beta \in B_n$, $\|\beta\|_A$ denotes the word length of β with respect to A_n (as A_n contains Σ_n, we have necessary $\|\beta\|_A \leqslant \|\beta\|_\sigma$ for all braid β of B_n). Hence Algorithm 4 running on w returns a word of length at most $(2n-3)\|\overline{w}\|_A$.

References

[1] J Birman, K. H Ko, and S. J. Lee, *A new approach to the word and conjugacy problems in the braid groups*, Adv. Math. **139** (1998), no. 2, 322–353.
[2] X. Bressaud, *A normal form for braids*, J. Knot Theory Ramifications **17** (2008), no. 6, 697–732.
[3] S. Burckel, *The wellordering on positive braids*, J. Pure Appl. Algebra **120** (1997), no. 1, 1–17.
[4] P. Dehornoy, *Braid groups and left distributive operations*, Trans. Amer. Math. Soc. **345** (1994), no. 1, 115–150.
[5] ———, *A fast method for comparing braids*, Adv. Math. **125** (1997), no. 2, 200–235.

[6] ———, *Alternating normal forms for braids and locally Garside monoids*, J. Pure Appl. Algebra **212** (2008), no. 11, 2413–2439.
[7] P. Dehornoy and L. Paris, *Gaussian groups and Garside groups, two generalisations of Artin groups*, Proc. London Math. Soc. (3) **79** (1999), no. 3, 569–604.
[8] I. Dynnikov and B. Wiest, *On the complexity of braids*, J. Eur. Math. Soc. (JEMS) **9** (2007), no. 4, 801–840.
[9] R. Fenn, M. T. Greene, D. Rolfsen, C. Rourke, and B. Wiest, *Ordering the braid groups*, Pacific J. Math. **191** (1999), no. 1, 49–74.
[10] J. Fromentin, *Every braid admits a short sigma-definite representative*, arXiv:0811.3902, to appear in J. Eur. Math. Soc.
[11] ———, *A well-ordering of dual braid monoids*, C. R. Math. Acad. Sci. Paris **346** (2008), 729–734.
[12] ———, *The well-ordering of dual braid monoid*, J. Knot Theory Ramifications **19** (2010), no. 5, 631–654.
[13] F. A. Garside, *The braid group and other groups*, Quart. J. Math. Oxford Ser. (2) **20** (1969), 235–254.
[14] D.M. Larue, *Left distributive and left-distributive idempotent algebras*, Ph.D. thesis, University of Colorado, Boulder, 1994.

Jean Fromentin
Univ Lille Nord de France, F-59000 Lille, France
ULCO, LMPA J. Liouville, B.P. 699, F-62228 Calais, France
CNRS, FR 2956, France
fromentin@lmpa.univ-littoral.fr

Luis Paris
Université de Bourgogne
Institut de Mathématiques de Bourgogne, UMR 5584 du CNRS,
B.P. 47870, 21078 Dijon Cedex, France
lparis@u-bourgogne.fr